

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  def cart2pol(x, y):
5      rho = np.sqrt(x**2 + y**2)
6      phi = np.arctan2(y, x)
7      return(rho, phi)
8
9  def pol2cart(rho, phi):
10     x = rho * np.cos(phi)
11     y = rho * np.sin(phi)
12     return(x, y)
13
14     #####
15     #initial conditions
16
17     x_0 = 10
18     y_0 = -50
19     v_int = 1/10
20
21     r_0 = [(x_0, 0)]
22     v_0 = [(0, v_int)]
23
24     r = r_0
25     v = v_0
26
27     h = x_0*v_int
28     p = h**2
29     E_0 = .5*v_int**2-1/x_0
30     a = -1/(2*E_0)
31     e = (1-p/a)**.5
32     P = 2*np.pi*a**(3/2)
33     t = P/1000
34
35     #####
36     #Exact Solution
37
38     theta_ex = np.arange(0, 2*np.pi+.01, 0.01)
39     r_ex = p/(1-e*np.cos((theta_ex)))
40
41     theta_comp = np.arange(0, 2*np.pi+.01, 0.1)
42     r_comp = p/(1-e*np.cos((theta_comp)))
43
44
45
46     #####
47     #Graphing
48     fig=plt.figure(1)
49
50
51     #Exact Solution
52     ax1=fig.add_subplot(221, projection='polar')
53     ax1.plot(theta_ex, r_ex)
54     ax1.set_rmax(.5)
55     ax1.set_rticks([3, 6, 9, 12]) # less radial ticks
56     ax1.set_rlabel_position(-22.5) # get radial labels away from plotted line
57     ax1.grid(True)
58     ax1.set_title("Kepler Solution", va='bottom')
59
60
61     #####
62     #Cromer Algorithm
63
64     for i in range(0,int(P*100)):
65         r_mag = (r[len(r)-1][0]**2+r[len(r)-1][1]**2)**.5
66         acc = np.asarray((-r[len(r)-1][0]/r_mag**3,-r[len(r)-1][1]/r_mag**3))
67

```

```

68     v.append(tuple(map(sum, zip(v[len(v)-1],acc*t))))
69     v_add = np.asarray(v[len(v)-1])
70     r.append(tuple(map(sum, zip(r[len(r)-1],v_add*t))))
71
72     r = np.asarray(r)
73     l = []
74
75     for i in r:
76         l.append(cart2pol(i[0],i[1]))
77
78     x_val = [x[0] for x in l]
79     y_val = [x[1] for x in l]
80
81
82
83     #####
84     #Graphing Cromer Algorithm
85
86     ax2=fig.add_subplot(222, projection='polar')
87     ax2.plot(y_val,x_val)
88     ax2.plot(theta_comp, r_comp, 'o', markerfacecolor='none', markeredgecolor='r')
89     ax2.set_rmax(.5)
90     ax2.set_rticks([3, 6, 9, 12]) # less radial ticks
91     ax2.set_rlabel_position(-22.5) # get radial labels away from plotted line
92     ax2.grid(True)
93     ax2.set_title("Cromer Algorithm", va='bottom')
94
95     #####
96     #Cromer Energy
97
98     r_mag = []
99     v_mag = []
100    E_t = []
101    time = []
102    timePeriod = []
103    E_rat = []
104    for i in r:
105        r_mag.append((i[0]**2+i[1]**2)**.5)
106    for i in v:
107        v_mag.append((i[0]**2+i[1]**2)**.5)
108    for i in range(len(r_mag)):
109        E_t.append(.5*v_mag[i]**2-1/r_mag[i])
110    for i in range(0,int(P*100)+1):
111        x = i*t
112        time.append(x)
113    for i in time:
114        x = i/P
115        timePeriod.append(x)
116    for i in range(len(E_t)):
117        E_rat.append(E_t[i]/E_0-1)
118
119    E_rat = E_rat[450:550]
120    timePeriod = timePeriod[450:550]
121
122    fig2, ax5 = plt.subplots()
123    ax5.set_ylabel('E(t)/E_0-1')
124    ax5.set_xlabel('Time/Period')
125    ax5.set_title("Energy Ratio")
126    ax5.plot(timePeriod,E_rat)
127
128    #####
129    #Runge-Kutta
130    x_0 = 10
131    y_0 = -50
132    v_int = 1/10
133
134    r_0 = [(x_0, 0)]

```

```

135 v_0 = [(0, v_int)]
136
137 r = r_0
138 v = v_0
139
140 for i in range(0,int(P*100)):
141     r_mag = (r[len(r)-1][0]**2+r[len(r)-1][1]**2)**.5
142     acc = np.asarray((-r[len(r)-1][0]/r_mag**3,-r[len(r)-1][1]/r_mag**3))
143
144     v_add = np.asarray(v[len(v)-1])
145     r_add = np.asarray(r[len(r)-1])
146
147     r.append(tuple(map(sum, zip(r[len(r)-1],v_add*t,.5*t**2*acc))))
148
149     r_2 = r_add+.5*t*v_add
150     r_mag2 = ((r_add[0]+.5*t*v_add[0])**2+(r_add[1]+.5*t*v_add[1])**2)**.5
151     acc_2 = np.asarray((-r_2[0]/r_mag2**3,-r_2[1]/r_mag2**3))
152
153
154     v.append(tuple(map(sum, zip(v[len(v)-1],t*acc_2))))
155
156 r = np.asarray(r)
157 l = []
158
159 for i in r:
160     l.append(cart2pol(i[0],i[1]))
161
162 x_val = [x[0] for x in l]
163 y_val = [x[1] for x in l]
164
165 #Graphing
166
167 ax3=fig.add_subplot(223, projection='polar')
168 ax3.plot(y_val,x_val)
169 ax3.plot(theta_comp, r_comp, 'o', markerfacecolor='none', markeredgecolor='r')
170 ax3.set_rmax(.5)
171 ax3.set_rticks([3, 6, 9, 12]) # less radial ticks
172 ax3.set_rlabel_position(-22.5) # get radial labels away from plotted line
173 ax3.grid(True)
174 ax3.set_title("Runge-Kutta", va='bottom')
175
176 #####
177 #Runge-Kutta Energy
178
179 r_mag = []
180 v_mag = []
181 E_t = []
182 time = []
183 timePeriod = []
184 E_rat = []
185 for i in r:
186     r_mag.append((i[0]**2+i[1]**2)**.5)
187 for i in v:
188     v_mag.append((i[0]**2+i[1]**2)**.5)
189 for i in range(len(r_mag)):
190     E_t.append(.5*v_mag[i]**2-1/r_mag[i])
191 for i in range(0,int(P*100)+1):
192     x = i*t
193     time.append(x)
194 for i in time:
195     x = i/P
196     timePeriod.append(x)
197 for i in range(len(E_t)):
198     E_rat.append(E_t[i]/E_0-1)
199
200 E_rat = E_rat[450:550]
201 timePeriod = timePeriod[450:550]

```

```

202 # fig2, ax5 = plt.subplots()
203 ax5.plot(timePeriod,E_rat)
204
205
206
207
208 #####
209 #Velocity Verlet
210 x_0 = 10
211 y_0 = -50
212 v_int = 1/10
213
214 r_0 = [(x_0, 0)]
215 v_0 = [(0, v_int)]
216
217 r = r_0
218 v = v_0
219
220
221 for i in range(0,int(P*100)):
222     r_mag = (r[len(r)-1][0]**2+r[len(r)-1][1]**2)**.5
223     acc = np.asarray((-r[len(r)-1][0]/r_mag**3,-r[len(r)-1][1]/r_mag**3))
224     v_add = np.asarray(v[len(v)-1])
225     r_add = np.asarray(r[len(r)-1])
226
227     r.append(tuple(map(sum, zip(r[len(r)-1],v_add*t,.5*t**2*acc))))
228
229     r_mag2 = (r[len(r)-1][0]**2+r[len(r)-1][1]**2)**.5
230     acc_2 = np.asarray((-r[len(r)-1][0]/r_mag2**3,-r[len(r)-1][1]/r_mag2**3))
231
232     v.append(tuple(map(sum, zip(v[len(v)-1],.5*t*acc,.5*t*acc_2))))
233
234 r = np.asarray(r)
235 l = []
236
237 for i in r:
238     l.append(cart2pol(i[0],i[1]))
239
240 x_val = []
241 y_val = []
242
243 x_val = [x[0] for x in l]
244 y_val = [x[1] for x in l]
245
246
247 #Graphing
248 ax4=fig.add_subplot(224, projection='polar')
249 ax4.plot(y_val,x_val)
250 ax4.plot(theta_comp, r_comp, 'o', markerfacecolor='none', markeredgecolor='r')
251 ax4.set_rmax(.5)
252 ax4.set_rticks([3, 6, 9, 12]) # less radial ticks
253 ax4.set_rlabel_position(-22.5) # get radial labels away from plotted line
254 ax4.grid(True)
255 ax4.set_title("Velocity Verlet", va='bottom')
256
257
258
259 #####
260 #Velocity Verlet Energy
261
262 r_mag = []
263 v_mag = []
264 E_t = []
265 time = []
266 timePeriod = []
267 E_rat = []
268 for i in r:

```

```

269     r_mag.append((i[0]**2+i[1]**2)**.5)
270 for i in v:
271     v_mag.append((i[0]**2+i[1]**2)**.5)
272 for i in range(len(r_mag)):
273     E_t.append(.5*v_mag[i]**2-1/r_mag[i])
274 for i in range(0,int(P*100)+1):
275     x = i*t
276     time.append(x)
277 for i in time:
278     x = i/P
279     timePeriod.append(x)
280 for i in range(len(E_t)):
281     E_rat.append(E_t[i]/E_0-1)
282
283 E_rat = E_rat[450:550]
284 timePeriod = timePeriod[450:550]
285 # fig2, ax5 = plt.subplots()
286 ax5.plot(timePeriod,E_rat)
287
288 plt.legend(('Cromer', 'Runge-Kutta', 'Velocity Verlet'), loc='upper right')
289 plt.show()
290 #####
291
292
293
294
295

```